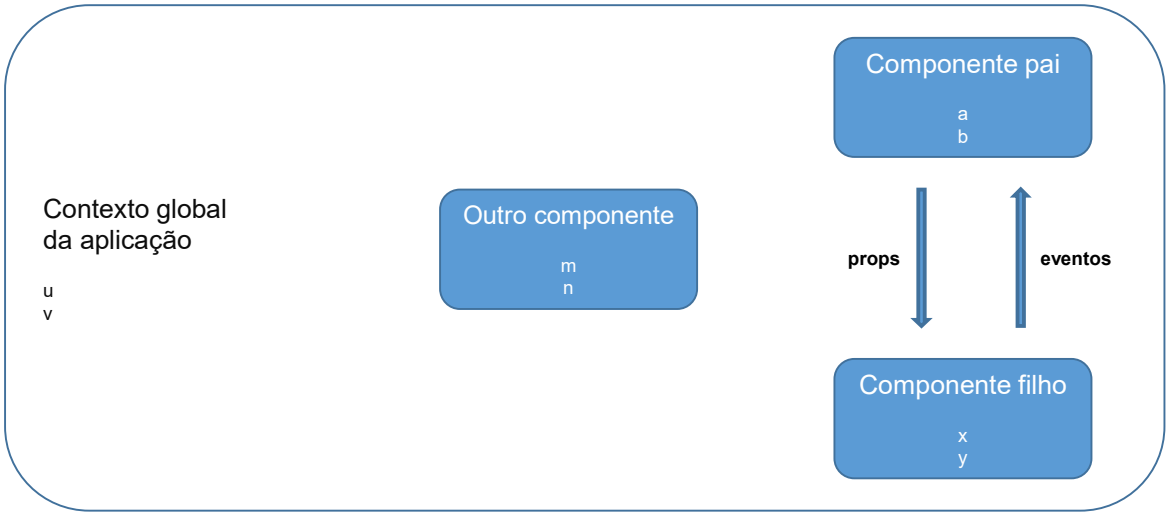
DEVSUPERIOR

Formação Desenvolvedor Moderno
Módulo: Front end

Capítulo: Eventos e estado global - comunicação entre componentes

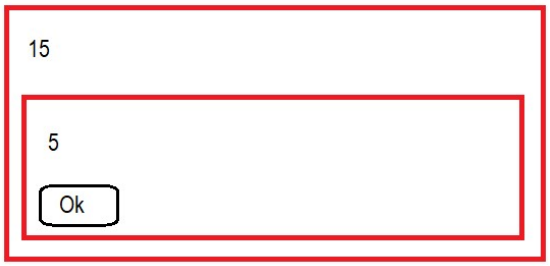
https://devsuperior.com.br

Comunicação entre componentes



Evento

Exemplo prático: queremos fazer um componente pai que mostra o triplo de um valor contido em seu componente filho. Sempre que o valor do componente filho mudar, o valor do componente pai também deve ser atualizado com o triplo do valor do filho.



Estratégia

- O componente pai deve passar uma função como argumento (prop) para o componente filho.
- Desta forma, o componente filho poderá fazer a chamada (ou ativação) da função do componente pai.
- Dizemos que:
  - O componente filho **envia um evento** para o componente pai
  - O componente pai **observa** o evento do componente filho, e **reage** a ele
- Note que o componente filho não sabe da existência do componente pai (desacoplamento).
- Padrão de nomes adotado:
  - Nome do parâmetro (prop) no componente filho: **on**Evento
  - Nome da função do componente pai passada como argumento: **handle**Evento

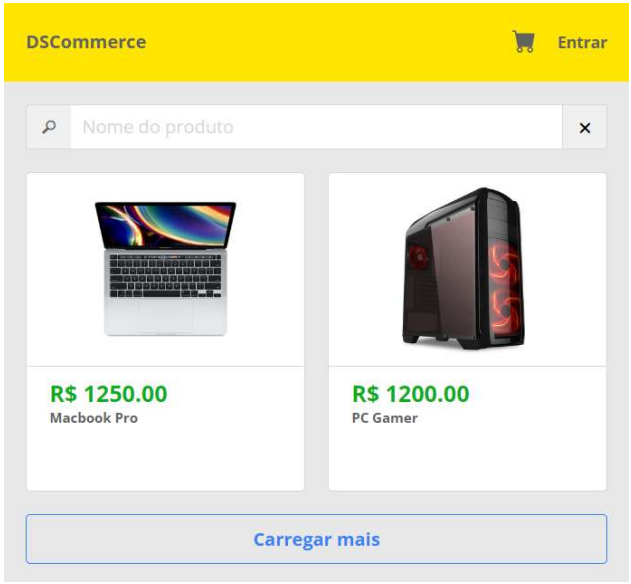
```tsx
import { useState } from "react";

type Props = {
    onNewValue?: Function;
}

export default function ChildComponent({ onNewValue }: Props) {

    const [count, setCount] = useState(0);

    function handleClick() {
        const newCount = count + 1;
        setCount(newCount);
        if (onNewValue) {
            onNewValue(newCount);
        }
    }

    return (
        <div style={{ border: "1px solid red", padding: "10px", }}>
            {count}
            <button onClick={handleClick}>
                Ok
            </button>
        </div>
    );
}
```

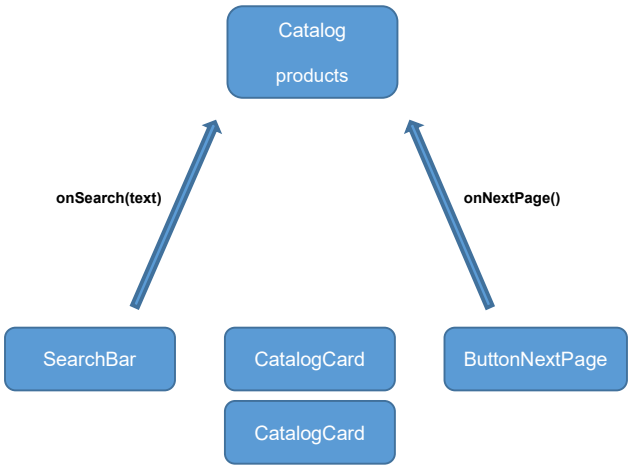

5

```tsx
import { useState } from "react";
import ChildComponent from "./filho";

export default function ParentComponent() {

    const [triple, setTriple] = useState(0);

    function handleNewValue(newValue: number) {
        setTriple(newValue * 3);
    }

    return (
        <div style={{ border: "1px solid red", padding: "10px", }}>
            {triple}
            <ChildComponent onNewValue={handleNewValue} />
        </div>
    )
}
```

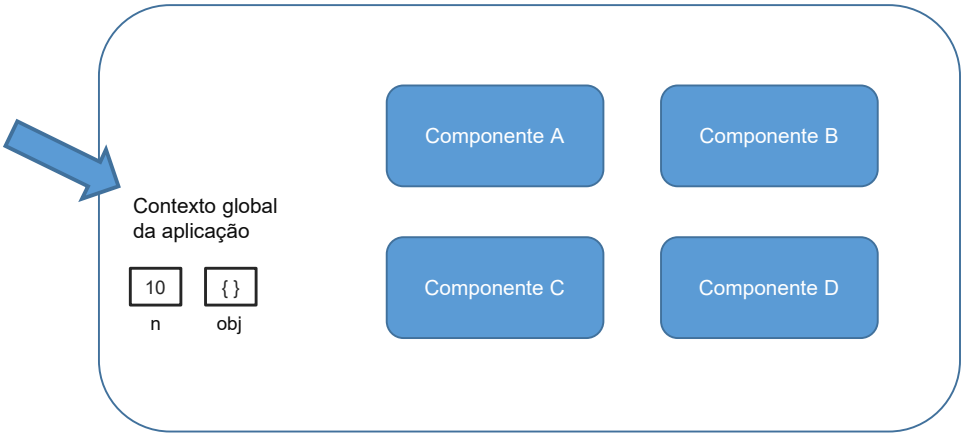

6

Exemplo prático (controles da listagem)





Estado global com Context API



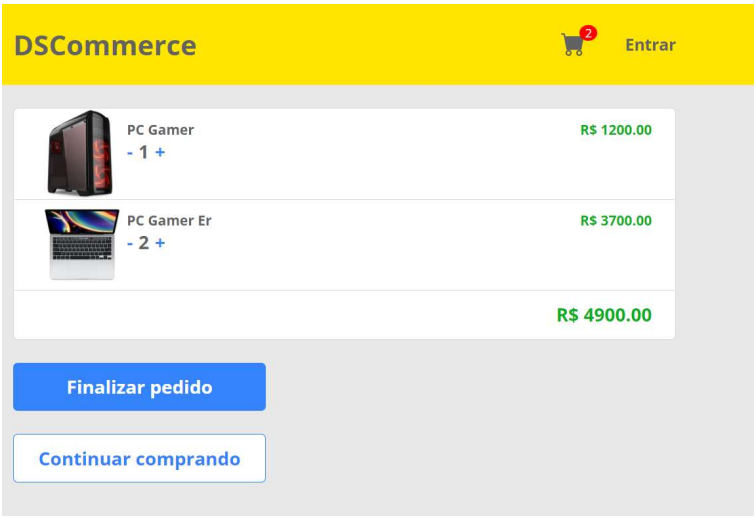
Mas o localStorage já não é um estado global?

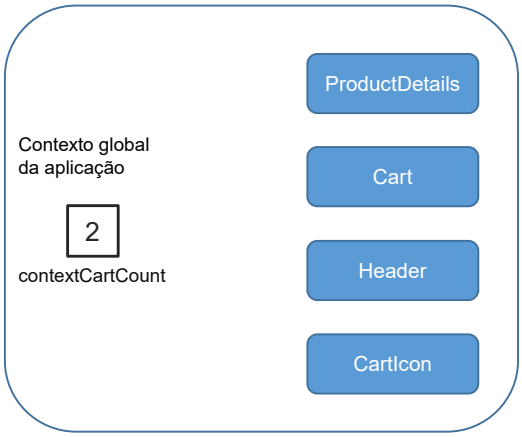
Sim, localStorage é uma forma de armazenar estado global.

Porém os componentes React não conseguem **observar** e **reagir** a mudanças no localStorage.

A Context API do React permite criar um estado (useState) global, de modo que qualquer componente possa observar e reagir às suas mudanças.

Exemplo prático (número de itens do carrinho)





Passos para usar um estado global

- Criar o contexto
  - Definir o tipo do contexto (dado + função set)
  - Criar o contexto com a função createContext
- Prover o contexto globalmente
  - Instanciar um useState em App.tsx
  - Prover o contexto usando o método Provider
- Acessar o estado global nos componentes

context-cart.ts

```
import { createContext } from "react";

export type ContextCartCountType = {
  contextCartCount: number;
  setContextCartCount: (cartCount: number) => void;
};

export const ContextCartCount = createContext<ContextCartCountType>({
  contextCartCount: 0,
  setContextCartCount: () => {}
});
```

App.tsx

```
export default function App() {

  const [contextCartCount, setContextCartCount] = useState<number>(0);

  useEffect(() => {
    const cart = cartService.getCart();
    setContextCartCount(cart.items.length);
  }, []);

  return (
    <ContextCartCount.Provider value={{ contextCartCount, setContextCartCount }}>
      <HistoryRouter history={history}>
        <Routes>
          ...
        </Routes>
      </HistoryRouter>
    </ContextCartCount.Provider >
  )
}
```

13

13

Nos componentes

```
export default function MyComponent() {

    const { contextCartCount, setContextCartCount } = useContext(ContextCartCount);


    ...
}
```

14

14